

# Save Plot to an Output File

Â  
Â

[Contents](#) [Previous](#) [Next](#)

**Goal:** Guide you through saving a plot in various graphics file formats.

Before running the tutorial below, type "*python*" or "*cdat*" at the command line.Â You will see the python prompt appear (i.e., ">>>"). You can now enter the command lines below.

You can [view](#) or [download](#) the full source code. To run the source code at the command line, type: "*python output\_file.py*".

```
# Import the modules needed for the tutorial
# cdms - Climate Data Management system accesses gridded data.
# vcs - Visualization and control System 1D and 2D plotting routines.
# cdutil - Climate utilitizes that contains miscellaneous routines for
#           manipulating variables.
# time - This module provides various functions to manipulate time values.
# os - Operation System routines for Mac, DOS, NT, or Posix depending on
#       the system you're on.
# sys - This module provides access to some objects used or maintained by
#       the interpreter and to functions that interact strongly with the interpreter.
import vcs, cdms, cdutil, time, os, sys

# Open data file:
filepath = os.path.join(sys.prefix, 'sample_data/clt.nc')
cdmsfile = cdms.open( filepath )

# Extract a 3 dimensional data set and get a subset of the time dimension
data = cdmsfile('clt', longitude=(-180, 180), latitude = (-90., 90.))

# Initial VCS:
v = vcs.init()
```

Get a boxfill and isofill graphics method objects and the template objects and plot:

```
# Assign the variable "bf_asd" to the persistent 'ASD' boxfill graphics methods.
bf_asd = v.getboxfill( 'ASD' )

# Assign the variable "cf_asd" to the persistent 'ASD' isofill graphics methods.
cf_asd = v.getisofill( 'ASD' )

# Assign the variables "bf_asd1" and "bf_asd2" to the persistent
# boxfill graphics methods.
tplt_asd1 = v.gettemplate( 'ASD1_of_2' )
tplt_asd2 = v.gettemplate( 'ASD2_of_2' )

# Plot the data using the above graphics methods and templates.
# Plot the data in background mode using the "bg=1" option.
# In this example two plots will be plotted on the VCS Canvas.
v.plot( data, bf_asd, tplt_asd1, bg=1 )
v.plot( data, cf_asd, tplt_asd2, bg=1 )
```

Generate a postscript output file.

```
v.postscript('test.ps')
```

Generate a gif output file.

```
v.gif('test.gif')
```

Generate a cgm output file.

```
v.cgm('test.cgm')
```

Generate a raster output file.

```
v.raster('test.ras')
```

Generate a ghostscript output file.

This routine allows the user to save the VCS canvas in one of the many GhostScript (gs) file types

If no path/file name is given and no previously created gs file has been designated, then file

  Â  /\$HOME/PCMDI\_GRAPHICS/default.gs

will be used for storing gs images. However, if a previously created gs file exist, then this out

By default, the page orientation is in Landscape mode (l). To translate the page orientation to p

The gs command is used to create a single gs file at this point. The user can use other tools to

Example of Use:

```
Â Â Â a=vcs.init()  
Â Â Â a.plot(array)  
Â Â Â a.gs('example') #defaults: device='png256', orientation='l' and resolution='792x612'  
Â Â Â a.gs(filename='example.tif', device='tiffpack', orientation='l', resolution='800x600')  
Â Â Â a.gs(filename='example.pdf', device='pdfwrite', orientation='l', resolution='200x200')  
Â Â Â a.gs(filename='example.jpg', device='jpeg', orientation='p', resolution='1000x1000')
```

Â Â Â

```
# generate ghostscript png output file  
x.gs('test.png')
```

```
# generate ghostscript tiff output file  
x.gs('test.tif', device = 'tiff24nc', orientation = 'l', resolution = '172.x172.')
```

Â

[Contents](#) [Previous](#) [Next](#)